PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 4 : G06F 9/46

A1

(11) International Publication Number: WO 87/ 02486

(43) International Publication Date: 23 April 1987 (23.04.87)

(21) International Application Number: PCT/US86/02018

(22) International Filing Date: 26 September 1986 (26.09.86)

(31) Priority Application Numbers: 787,668
787,669
787,781

(32) Priority Dates: 15 October 1985 (15.10.85)
15 October 1985 (15.10.85)
15 October 1985 (15.10.85)

(33) Priority Country: US

(71) Applicant: BURROUGHS CORPORATION [US/US]; Burroughs Place, Detroit, MI 48232 (US).

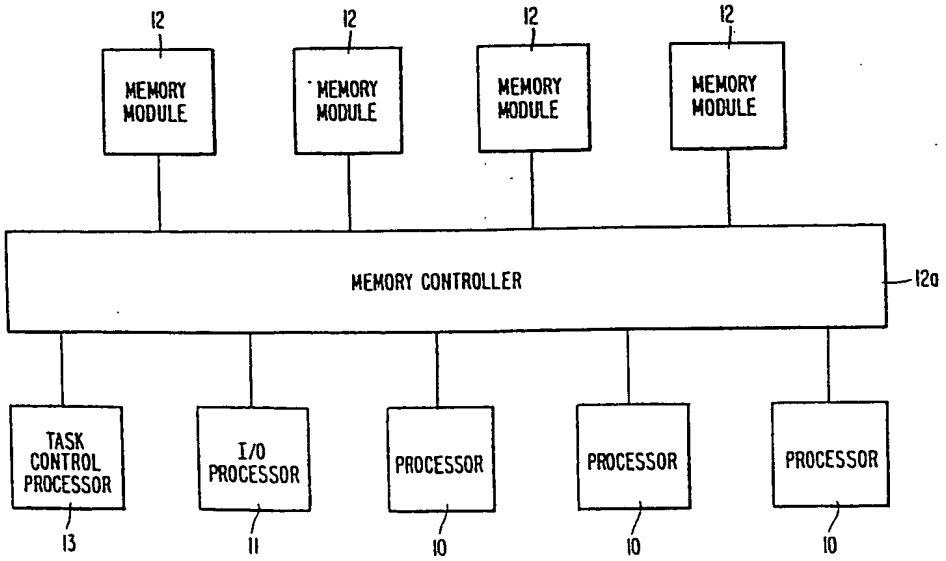
(72) Inventors: JENNINGS, Andrew, Thomas ; 701 Green Ave., West Chester, PA 19380 (US). KELLER, John, Allen ; 12 Valley Green Dr., Coatesville, PA 19320 (US).

(74) Agent: PETERSON, Kevin, R.; Burroughs Corporation, Burroughs Place, Detroit, MI 48232 (US).

(81) Designated States: AT (European patent), BE (European patent), CH (European patent), DE (European patent), FR (European patent), GB (European patent), IT (European patent), JP, LU (European patent), NL (European patent), SE (European patent).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: A SPECIAL PURPOSE PROCESSOR FOR OFF-LOADING MANY OPERATING SYSTEM FUNCTIONS IN A LARGE DATA PROCESSING SYSTEM



(57) Abstract

A task control mechanism (13) for maintaining a queue of ready or available processes linked together according to an assigned priority for a plurality of central processors (10) where the processors (10) may be assigned to the highest priority task when that processor is not busy executing some higher priority task. The task control mechanism (13) also includes a mechanism (26) for computing task priorities as new task are inserted into the queue or removed. The mechanism (13) also maintains an event table (20a) which is really a table of event designations to be allocated to different processes upon request where the requesting processes assign a particular function or 'meaning' to the event designation. The mechanism (13) of the present invention maintains the state of such allocated events in the event table (20a) and signals the related (or 'waiting') processes that an event has happened so that the particular system central processors (10) assigned to execute those particular processes may then proceed with their execution.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FR	France	ML	Mali
AU	Australia	GA	Gabon	MR	Mauritania
BB	Barbados	GB	United Kingdom	MW	Malawi
BE	Belgium	HU	Hungary	NL	Netherlands
BG	Bulgaria	IT	Italy	NO	Norway
BJ	Benin	JP	Japan	RO	Romania
BR	Brazil	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SN	Senegal
CH	Switzerland	LI	Liechtenstein	SU	Soviet Union
CM	Cameroon	LK	Sri Lanka	TD	Chad
DE	Germany, Federal Republic of	LU	Luxembourg	TG	Togo
DK	Denmark	MC	Monaco	US	United States of America
FI	Finland	MG	Madagascar		

A SPECIAL PURPOSE PROCESSOR FOR OFF-LOADING MANY OPERATING SYSTEM FUNCTIONS IN A LARGE DATA PROCESSING SYSTEM

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to a special purpose processor and more particularly to such a processor for off-loading many operating system functions that would otherwise be executed by one or more central processors in a large data processing system.

Description of the Prior Art

Multi-processing systems and a single processing

system adapted for multi-programming require operating systems for the purpose of scheduling tasks on a particular processor, initiating input/output data transfers, handling external interrupts, and the like. Such operating systems, sometimes called a master control program (MCP), actually consist of many different particular processes and subroutines, each of which has a particular function. Such particular processes and subroutines reside in main memory and currently must be executed by the particular central processors when a processor or another central processor in a multi-processing system requires service, such as when it has finished a particular process or task, requires an input/output operation or some other service.

More specifically, among other things, the operating systems allocate and deallocate events where an event may be an external interrupt, I/O operation, etc.; implement a set of functions which may be performed upon these events; maintain the status of processes or tasks running on the system; perform task priority computations and schedule the execution of tasks by various central processors; maintain the system timers, including the interval timers; and perform some accounting and billing functions.

Statistical studies indicate that a major portion of each processor's time, in a multi-processing system, is employed in executing operating system functions. From these studies, it is estimated that the overhead of such management functions has been anywhere between ten percent and fifty percent, and occasionally even higher. Furthermore, a goodly portion of the time that the corresponding central processor is executing operating system functions is employed in establishing process priority, performing functions on events (as defined above) and initiating input/output operations.

If even these latter functions could be removed from the operating systems, then the through-put of the data processing system should be substantially enhanced.

It is then the object of the present invention to provide a data processing system having improved through-put.

It is another object of the present invention to provide an improved data processing system wherein those operating system functions which require most of the central processor time are removed from the main memory of the system.

It is still a further object of the present invention to provide an improved data processing system having facilities for performing those functions that would otherwise be a part of the operating systems stored in memory.

SUMMARY OF THE INVENTION

In order to accomplish the above-identified objects, the present invention resides in a special purpose processor for a large data processing system which special purpose processor performs many of the functions of the system operating systems that utilize most of the central processor's running time when those processors are executing routines of the operating systems stored in main memory. More specifically, the basic functions of the special purpose processor are that of process or task scheduling and the allocation of events to such processes or tasks, which events are requested by or affect the execution of the individual tasks.

Particularly, such a processor maintains a queue of ready or available processes linked together according to an assigned priority so that any central processor may be assigned to the highest priority task when that processor is not busy executing some higher priority task. The special

purpose processor also includes a mechanism for computing task priorities as new tasks are inserted into the queue or removed.

The processor of the present invention also maintains an event table which is really a table of event designations to be allocated to different processes upon request where the requesting processes (including the MCP) assign a particular function or "meaning" to the event designation. The processor of the present invention maintains the state of such allocated events in the event table and signals relocated (or "waiting") processes that an event has happened so that the particular central processors assigned to execute those particular processes may then proceed with their execution.

A feature then of the present invention resides in a special purpose processor for a large data processing system which processor maintains a queue of ready processes linked together in the order of their assigned priority so that they may be executed by the various central processors of the system as those central processor become available and also an events allocation mechanism for allocating available events to the various processes upon request.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other objects, advantages and features of the present invention will become more readily apparent from a review of the following specification when taken in conjunction with the drawings wherein:

FIG. 1 is a diagram of the large data processing system employing the present invention;

FIGS. 2A-D are diagrams representing a portion of memory containing data arranged as push-down stacks and also a plurality of display registers used to access various elements within the stack;

FIG. 3 is a schematic diagram of the special purpose processor of the present invention;

FIGS. 4A and B are respectively a schematic of the process table and also the four-word format for a particular process which four-word blocks are stored in the process table;

FIGS. 5A-D are formats of the four words representing an event as stored in the event table of the present invention;

FIG. 6 is a schematic diagram illustrating the inputs to the arithmetic logic unit (ALU) of the processor of the present invention;

FIG. 7 is a schematic diagram illustrating the inputs and outputs of the process table of the present invention;

FIG. 8 is a schematic diagram of the event table support logic; and

FIGS. 9A and B are diagrams illustrating the manner in which processes are linked in a queue according to priority when waiting to procure an event and how processes waiting upon the same event to happen are so linked.

GENERAL DESCRIPTION OF THE INVENTION

A large data processing system, and in particular a multi-processing system, which employs the present invention is illustrated in FIG. 1. The system includes a plurality of main processing units 10 and one or more I/O processors 11, each of which can communicate with a plurality of memory modules 12, by way of memory controller 12a. The present invention is an addition to the system in the form of task control processor 13 which also communicates with any one of processors 10 by way of controller 12a.

While the present invention may be used in a conventional system of sequential or von Neumann type processors, the preferred embodiment of the present invention, as described below, is directed toward a system of processors designed for the execution of block-structured programming languages, i.e., nested declarations, which is a natural form for the expression of complex algorithms. A particular processor that was designed to employ such block-structured, or nested languages, is described in the Barton et al. U. S. Patents No. 3,461,434; 3,546,677 and 3,548,384. These patents describe a stack-oriented data processor where the stack mechanism, a first-in last-out mechanism, handles the flow of operators and associated parameters in a manner which reflects the nested structure of the particular higher level languages employed. Such languages include ALGOL and ALGOL-type languages, such a PL/1, EULER, etc. (Cf., E. I. Organick Computer System Organization, Academic Press, 1973).

A system of the type described in the above-identified Barton patents is oriented around the concept of a segmented memory and specially treated segments called stacks. The processor runs in an expression stack; operators take their arguments from the top of the stack and leave their results on the top of the stack. The data addressing space of the executing program is mapped into the stack as well as other stacks linked to it and data segments referenced by the descriptors contained in the stack structure.

The addressing environment of the executing code stream consists of a set of local addressing spaces contained within the stacks. These are called activation records or lexical regions and each consists of a set of variables addressed by an index relative to the base of the activation

record. That is to say, addressing of a given data item is by way of an address couple of the form (Lambda, Delta) where Lambda is the lexical level of a given activation record in the stack and Delta is the offset to the variable from the base of the activation record at level Lambda. In order to access any activation record within a stack, the respective records, or lexical regions, are linked together by pointers from the base of the topmost activation record to the lowest level activation record. In the above-described Barton patents, addressing is optimized by defining an array of "display" registers in the processor which registers are maintained in such a manner that element i in the array contains the base of the activation record at the level i.

A data stack as might exist in one of memory modules 12 of FIG. 1 is illustrated in FIG. 2A and consists of four activation records at lexical levels 0-3, where the value of the topmost lexical level is stored in a lexical level register in a corresponding processor. The actual addresses in memory of the respective bases of the activation records are shown in 2A and these addresses are stored in the display registers of the corresponding processor in a manner illustrated in FIG. 2B.

Activation records are created by the execution of a procedure entry operator by the processor. Thus, for the purposes of illustration, FIG. 2C illustrates that the processor is now working in a different stack or portion of memory. As a result, the display registers of a particular processor have had to be updated and this update is shown by the new contents of the display registers as shown in FIG. 2D.

Referring back to FIG. 1, the task control processor 13 is designed to relieve the master control program of many of its most time consuming functions. In the prior art, the respective central processors 10 of FIG. 1 would be running various programs or processes. That is to say, they would be operating in different stacks in memory. If, for example, a particular processor executed a branch instruction which called for a branch to another task, it would notify the master control program which would then initiate the required process and cause the requesting process to be put in a waiting state. When the required process has been finished, the master control program would then notify the requesting process that the required process was finished and the requesting process would then be reactivated. Other examples of master control program functions have been discussed above, such as handling of external interrupts and initialization of input/output operations. In addition, the master control program in prior art systems would handle the scheduling of various tasks for execution by the various processors 10 of FIG. 1 according to which processes had the highest priority.

The task control processor 13 of FIG. 1 is illustrated in more detail in FIG. 3. The two principal functional elements shown therein are process table 21 and event table 20a. Process table 21 and process statistics table 20b contain the information as to the status of all tasks or processes scheduled to be run on the system of FIG. 1. In the described embodiment of the present invention, there can be 4 K such tasks or processes running on the system at any one point in time.

The status information of the processes in process table 21 are arranged as a queue or a linked list of processes according to the priority of the processes involved.

During the operation of the system of FIG. 1, certain processes may be completed and removed from the linked list within process table 21 and new processes may be inserted. In each case, the task control processor of FIG. 3 computes the new priority ratings and rearranges the linked list. In this manner, the highest priority process is always ready to be assigned to an available processor 10 of FIG. 1 as required.

It should be evident from the description thus far that the terms "task", "process" and "stack" are used synonymously where a stack is a natural physical location in main memory and the respective tasks or processes are independent of one another and occupy the corresponding stack space. Similarly, the terms "stack number", "task number" and "process number" are used synonymously and are the actual addresses to process table 21 of FIG. 3 of the corresponding process status information.

Event table 20a is employed to contain information as to the status of various event designations called for by processes running on the system. In the described embodiment of the present invention, there may be a maximum of 512 K such events being utilized at any one time. When a process being executed by a particular processor 10 of FIG. 1 requires an event designation, it requests the allocation of such a designation from the task control processor of FIG. 3, which then allocates an unallocated event designation to that process and sends an event token to be placed in main memory on the top of the particular stack whose process requested that event designation. Event table 20a then upgrades the event information to indicate that the event has been allocated. The event token is made up of the event address to event table 20a and also certain coded bits to ensure that one of processors 10 of FIG. 1 does not accidentally create

its own event token. Event table 20a is also employed to maintain a linked list of various processes requesting a particular event that has already been allocated and assigns that event to the highest priority process requesting that event when the event is freed or liberated by its owning process.

As was indicated above, an event designation does not specify the particular function for which the event was allocated. This is done by the requesting process. Event table 20a serves the purpose of maintaining the status of the event, e.g., whether it is available for allocation, whether it has occurred, what processes are waiting on it, and so forth.

Continuing on with a description of FIG. 3, support logic 22 is employed to insert information fields into event table 20a, statistics table 20b and link table 20c as well as to extract fields therefrom as required. Local memory 23 serves as an output buffer and also maintains a processor table which indicates which processes are currently running on the respective processors 10 of FIG. 1.

Message transmission to the other processors of FIG. 1 are by way of memory controller 12a from output register 29 of FIG. 3. Messages are received from controller 12a by way of input register 25 to message buffer 24. As indicated in FIG. 3, the various functional units thus described have inputs to arithmetic logic unit module 26 by way of arithmetic logic unit input multiplexer 27. Arithmetic logic unit module 26 is employed to compute process priorities as described above and also to form messages for transmission to the other processors of the system. Clock timer 28 supplies real time clock values to arithmetic logic module 26 to aid in the computation of how long a particular process in a wait state has been waiting as

compared to the maximum amount of time it can wait before its execution must be reactivated (as well as statistics, and task scheduling algorithms).

All of the functional units of FIG. 3 are under the control of sequence control store 30 and are activated by the receipt of an external processor request by message buffer 24, where the request command is decoded by control store 30.

When the command received from the external processor is either a WAIT or PROCURE command, the special purpose processor of the present invention responds by transmitting to that processor a MOVE-STACK command which causes the processor to begin executing a different process assigned by the special purpose processor of the present invention. This command is stored in local memory 23 of FIG. 3 and is transmitted to the processor which made the initial request along with the process number or stack number of the next available process having the highest priority for execution as will be more thoroughly described below.

The task control processor of FIG. 3 serves to maintain the status of the various processes and control the execution of those processes by the respective central processors 10 of FIG. 1. A process may be in one of four main states as follows:

Alive	- currently executing on a processor;
Ready	- in a suitable state to run when a processor becomes available;
Waiting	- cannot be run until some event occurs; and
Selected	- A main processor has been instructed to move to this process stack and acknowledgment to this command has yet to be received by the task control processor.

In addition, the task control processor maintains two further state bits for each process, namely "scheduled" and "blocked". A scheduled process is one which is included in the scheduling mechanism and unscheduled when the process is to be removed from that mechanism. The four states listed above apply only to scheduled processes. A blocked process is one which will not be scheduled for execution on a processor until such time as it becomes "unblocked".

Process table 21 and statistics table 20b of FIG. 3 maintain the following information for each process in the system: the process current state, i.e., ready waiting, etc.; the process priority; accumulated processor and ready time; the processors to which the process may be assigned; maximum permitted processor time; and the class of service of the process.

The process class is used to implement a resource sharing scheme between different groups of processes. The task control processor of FIG. 3 also maintains system wide statistics on processor utilization and processor time consumed by processes of different classes. These process attributes may be set and read by software at any time, regardless of whether the process is currently under the control of the task control processor.

The task control process or FIG. 3 responds to process commands from the respective main processor. A SET-PRIORITY command changes the priority of a designated stack to a supplied value. A SET-CLASS command sets the statistics class of the designated stack. A SET-PROCESSOR command marks the designated stack as being executable only on a designated main processor. The INSERT command causes the designated stack to be included in the scheduling mechanism and the REMOVE command causes the designated stack to be deleted from the scheduling mechanism. The above

commands are merely examples of the types of commands issued to the task control processor to control process scheduling and many other commands are employed but their description is not required here.

The various commands described above are initiated by the main processors when they decode specific instructions in the code stream of the process that the respective processor is then executing. The same is true of the request for event allocation in processing. It will be remembered that when a given process or task requires an event to be allocated, this can only be communicated to the task control processor by one of the main processors executing that process.

The general nature of an event has been well described above. Access to an event is obtained by a particular process upon execution of an ALLOCATE command which, in return, requests the event from the task control processor of FIG. 3. The task control processor then generates an event token for an available event selected from event table 20a, which token is placed on top of the stack of the requesting process. Often the communications between the task control processor and the requesting processor are in terms of these event tokens.

In addition to the ALLOCATE function, other functions performed by the task control processor in response to a received command, include DEALLOCATE which causes the designated event to be returned to the pool of available events and there may be other processes which are either waiting on or attempting to procure the event in which case the requesting processor will be notified. The PROCURE command results in an attempt to procure a designated event on behalf of the process which is currently executing on the requesting processor. If the event is available, it will be

marked as unavailable, the owner will be designated as the stack number is currently executing. If the event is unavailable, the process will be suspended, the base priority of the owning process will be adjusted such that it is at least as great as the contending process' base priority and the requesting processor will be rescheduled.

The LIBERATE command causes the task control processor to pass ownership of the designated event to the highest priority process which has attempted to procure it. If no such process exists, the event is marked as available.

The SIMPLE-WAIT command suspends execution of the process currently executing on the requesting processor until the designated event has happened.

The MULTIPLE Wait command suspends execution of the process currently executing on the requesting processor until either a time limit has expired or one of a set of events has happened.

The CAUSE command will "awaken" all processes which are currently waiting on a designated event. The event will be marked as happened unless either a reset option is used or there is some stack performing a particular function upon the event, in which case it will be left in the not-happened state. If the event is available and has contenders, the highest priority contender will be given ownership of the event.

The SET command marks the designated event as happened without reactivating any process which may be waiting on it.

All of the tables of FIG. 3 are initialized whenever a halt-load is performed on the system. After that the tables are not reloaded but are modified from time to time depending upon the commands the task control processor is executing.

The assignment of the various processors 10 of FIG. 1 to the various processes or stacks in main memory depends on the commands sent to the respective processors under control of control store 30 of FIG. 3 which causes the respective processors to begin execution in various designated stacks. Thereafter, control store 30 of FIG. 3 will cause individual processors to be rescheduled (moved to another stack) primarily when that processor has executed a wait-on-event command or a procure command as was discussed above. Control store 30 of FIG. 3 will also cause a processor to move to another stack whenever it determines that it has been operating in its current stack for more than a prescribed time.

DETAILED DESCRIPTION OF THE INVENTION

FIG. 4 is a representation of the organization of process table 21 of FIG. 3 which is a static RAM with a capacity of 16K word locations of 17 bits each. This RAM is divided into 4K four-word blocks so as to provide the information required for the scheduling and manipulation of information for 4K processes as was described above. Each four-word block is addressed by a 12 bit process number, as was described above, which is supplied with 2 extra bits to select which of the words is to be addressed in the particular four-word block.

FIG. 4B illustrates the format of one of the four-word blocks in FIG. 4A.

In WORD-0 of FIG. 4B, bit 16 is the selectable bit which indicates that the process is a candidate for selection as head of the ready queue (which is described below). The process must be scheduled and ready and not blocked to be in this state. Bits 15-12 identify the statistics class to which class the current process belongs. Bits 11-0 are the ready queue forward link, which is used when chaining through

the ready queue in order of descending priority. That is to say, bits 11-0 are the 12 bit process number or address to the process table of the next lower priority process in the linked list of such processes.

Word-1 of FIG. 4 is similar to WORD-0 except that bits 11-0 now represent the ready queue reverse link or the process number of the next highest priority process in the linked list of processes scheduled on the system.

In WORD-2 of FIG. 4B, bit 16, when set, indicates that this process is at the head of the ready queue for its particular processor designation as will be described in relation to WORD-3. Bits 15-8 of WORD-2 are the most significant portion of the total priority of the process which priority is setable by some processor task and is manipulated by the task control processor of the present invention. Bits 8-0 represent the least significant portion of the total priority of the process and are manipulated by the scheduling algorithm employed by the task control processor of the present invention.

In WORD-3 of FIG. 4B, bit 16, when set, indicates whether or not the current process has been scheduled on the system. Bits 15-8 represent the original base priority of the process. This is the lowest base priority which the process would fall back to when lock contention conditions are resolved. Bits 7 and 6 indicate the type of processor upon which the current process is to be run (namely whether the processor is a main processor 10 of FIG. 1 or I/O processor 11 of FIG. 1, or a future processor type not yet implemented). Bits 5 and 4 represent the process state which may be Waiting, Ready, Selected (as was described above), and Alive. Bit 3, when set, indicates that the process has been "blocked" and task control processor will not schedule it for execution on a processor. Bit 2, when set, indicates that

this process is to be interrupted on the next move stack command telling a processor to move to this stack number.

Event table 20a of FIG. 3 contains storage locations of the status of 512K events. Like process table 21, each storage location in event table 20a is made up of a four-word block. The general formats of the respective words are illustrated in FIGS. 5A-D. Each block in the event table is addressed by an event number which is part of the event token, as was described above, along with two other bits which specify which of the four words is to be addressed. Each of the four words is 52 bits in width.

FIG. 5A represents the general format of WORD-0. As illustrated therein, there are two principal fields that are relevant to this disclosure. Bits 31-24 indicate the priority of the highest priority process which is contending or trying to procure this event while it is owned by some other process for reasons that were described above. Bits 18-0 contain the event number of the next lower priority event owned by the process which owns the current event. That is to say, this field is a forward link in a queue of all events currently owned by the owner of the current event.

FIG. 5B represents the general format of WORD-1 which is similar to that of WORD-0 except that bits 18-0 now represent the owner queue reverse link or the event number of the next highest priority event owned by the owner of the current event.

FIG. 5C generally represents the format of WORD-2 which has two fields that are of particular interest here. Bits 35-24 contain the number of the current owning process. Bits 11-0 also represent a process number which represents the highest priority process which is attempting to procure ownership of the current event.

FIG. 5D is a general format of WORD-3. Bit 51 of this word, when set, indicates that the current event has been allocated. Bit 50, when set, indicates that the event is available. Bit 49 is a special bit which, when set, indicates that ownership of that event has been obtained by a particular authorized process and taken away from all other processes that had had ownership. Bit 48, when set, indicates that the event has happened or occurred. Bit 47, when set, indicates that there are contenders or processes attempting to procure this event. Bit 46, when set, indicates that there are processes waiting for this event to happen. There are two other fields or sets of bits which are of significance to this disclosure. Bit 19-0 are a pointer or link to a list of processes which are waiting for the event to happen, which list is in link table 20c of FIG. 3. Bits 12-0 of this link are the process number of the first process in the list.

WORDS 0, 1 and 2, as described above, are basically concerned with processes waiting to procure events while WORD-3 is basically concerned with processes which are waiting on events. In regard to processes waiting on an event, it should be emphasized that only one process may own an event, or a plurality of events, at any one time. However, a number of processes can be waiting on a given event even though they don't own it. The reason for this is in the nature of the block-structured programming languages employed in the preferred embodiment of the present invention, as was discussed above in relation to FIGS. 2A-D. That is to say, with such block-structured languages, any process, although independent, is part of a hierarchy of processes. Thus, when a given process requests and is allocated an event, it notifies its parent process that it owns that event and has assigned a function to it and the

parent process in turn notifies other sibling processes that the particular event has been allocated and assigned a function. Any of the sibling processes requiring that event are then placed in a waiting state until the event occurs.

FIG. 6 is a more detailed schematic diagram of arithmetic logic unit module 26 and arithmetic logic unit input multiplexer 27 of FIG. 3. In FIG. 6, arithmetic logic unit module 26 includes arithmetic logic unit 40 which receives inputs from B register 41 and accumulator 42. The output of arithmetic logic unit 40 is to rotator 43 which is a shifting mechanism that may shift left end around either eight bits or one bit. The output of rotator 43 is to accumulator 42. The output of B register 41 and accumulator 42 are also supplied to output multiplexer 44. The input to arithmetic logic unit module 26 is by way of the series of input multiplexers 27a and 27b which together form arithmetic logic unit input multiplexer 27 of FIG. 3. The inputs to these two multiplexers were described above in regard to FIG. 3.

FIG. 7 illustrates the input and output logic for process table 21 of FIG. 3. In FIG. 7, the actual process table is process RAM 50 which is a 16K by 18 bits (including parity) static random access memory. Addresses and data are received from arithmetic logic unit module 26 of FIG. 3 with the addresses going directly to process RAM input multiplexer 51 and the data going directly to the input of process RAM 50. Process RAM input multiplexer 51 selects either the address from the arithmetic logic unit module or from the output of process RAM 50 which address is a 12 bit process number as was described above. The selected address is then sent to address counter 52 for addressing process RAM 50. Two bits can also be received from sequence control store 30

of FIG. 3 by word register 53 of FIG. 7. As was explained above, these two bits specify which word in a particular process word-block is to be selected. These two bits may also come from a constant which selects the priority word of a particular process block and word select multiplexer 54 selects between either the constant or the contents of word register 53. These two bits are then added to the 12 bit output of address counter 12 to create a 14 bit address to process RAM 50.

Magnitude comparator 56 serves to compare the priority of the current process block being addressed in process RAM 50 which a target priority as received by target priority register 57 from arithmetic logic unit module 26 of FIG. 3. This target priority represents the priority of a task to be inserted in the linked list of tasks being maintained in process RAM 50.

Selectable task function unit 58 of FIG. 7 serves to compare the class of each of the processes in process RAM 50 with the contents of class enable mask 59 to see which of the current processes are enabled for execution on an available processor, in which case sequence control store 30 of FIG. 3 is notified by selectable task function 58. Mask control function unit 60 serves to set various bits in class enable mask 59 to indicate which classes can be run on currently available processors.

FIG. 8 illustrates in more detail support logic 22 of FIG. 3 for receiving data for input into and fetching data from event table 20a, statistics table 20b and link table 20c of FIG. 3. All data transfers into and out of those tables is by way of staging register 70 which can receive data from the arithmetic logic unit module 26 of FIG. 3 or from respective tables themselves which are formed of dynamic RAMs. Staging register 70 can also receive data from its

output and these three inputs are selected by input select unit 71. The data remains in staging register 70 for one clock period while its parity or error correction code (ECC) is checked by check/generator 74. The data may then have fields extracted from it by field extraction unit 73 for transmission back to the arithmetic logic unit module or the entire output data from staging register 70 can be combined with a parity bit from check/generator 74 for transmission to the respective event table, link table or statistics table. The input to field extraction unit 73 is selected by field extract input multiplexer 72.

Addresses for the respective tables are generated by address formation function unit 75 by receiving a 20 bit field from field extraction unit 73. An 8 bit literal from sequence control store 30 of FIG. 3 informs address formation function unit 75 of the particular table to be addressed and formation function unit 75 then forms the appropriate address for transmission to address counter 76 from which it is sent to the respective table by address source multiplexer 78 which may also select an address from refresh address counter 77 when the dynamic RAMs of tables 20a, 20b and 20c of FIG. 3 are being refreshed.

FIG. 9A is a diagram of how ownership is passed from one process to the next lowest process waiting to procure an event. As shown therein, when an event becomes available having been freed by its owning process, the P-head field of WORD-2 of the particular event block is used as a pointer to link table 20c of FIG. 3 where the process number of the next lower priority process requesting ownership resides. The event is then assigned this process as owner and this process is then made ready for the next available processor 10 of FIG. 1. When that processor becomes available, it is given instruction to move to the stack

number of the now owning process. When the event again becomes available, the above procedure is then repeated.

FIG. 9B illustrates how link table 20c of FIG. 3 is employed to link processes waiting on this same event. As shown therein, when an event has occurred and its "happened" bit has been set, (due to the receipt of a CAUSE command received by the task control processor of FIG. 3), the W-head of WORD-3 of the particular event block is employed to point both to the particular process block in process table 21 and also to the process link in link table 20c and this action ripples through the link table and the process table such that all processes waiting on that event are made ready for the next available processor 10 of FIG. 1.

One advantage, among others, of the processor of the present invention is that external interrupts now vanish and the device requesting the interrupt, such as an I/O processor, merely signals the special purpose processor of the present invention that a specific event has occurred by sending a CAUSE command which causes that event to have its status changed to "has occurred".

EPILOGUE

A special purpose processor has been described above for the purpose of off-loading those operating system functions which consume most of the processing time of the various processors in a large data processing system. Specifically, the special purpose processor is adapted to schedule processes or tasks on the various processors as the processors become available, which processes are scheduled according to their assigned priority. The special purpose processor also maintains an event table which is really a table of event designations to be allocated to different processes upon request when the requesting processes (including the operating system) assign a particular function

to that event designation. The event table is then used to maintain the status of all such allocated events and to signal all processes waiting on a particular event that the event has happened.

While but one embodiment of the present invention has been disclosed, it will be apparent to those skilled in the art that variations and modifications may be made therein without departing from the spirit and the scope of the invention as claimed.

What is claimed is:

1. In a processing system having at least one central processor for executing processes and at least one memory module coupled to said central processor for storing a number of different process codes, a process scheduling mechanism comprising:

input means coupled to said at least one central processor to receive a command indicating that the process currently being executed by said central processor is in a wait state;

process table means for storing process designations of processes scheduled to be executed by said system including processes which are available to be executed and arranged according to assigned priorities;

output means coupled to said at least one central processor to transmit a move command to move execution to a different process in said at least one memory module; and

control means coupled to said input means, process table means, and output means to cause said output means to transmit said move command along with the process designation of the next highest priority available process.

2. A process scheduling mechanism according to claim 1 wherein:

said control means includes a control store to store sets of control signals for transmission to said input means, process table means and output means to effect their operation.

3. A process scheduling mechanism according to claim 1 wherein:

said output means includes a local memory as an output buffer which memory stores said move to new process command.

4. A process scheduling mechanism according to claim 1 wherein said input means further includes:

message buffer means to store a plurality of commands from said at least one central processor.

5. A process scheduling mechanism according to claim 1 further including:

logic unit means coupled to said control means and said process table means to change process state designations in said process table means as the various processes change from the being executed state to a wait state or a ready state.

6. A process scheduling mechanism according to claim 1 further including:

priority computation means coupled to said process table means and to said control means to change the priority of various process designations in said process table means when new processes are scheduled for execution by said processing system.

7. In a processing system having at least one central processor for executing processes and at least one memory module coupled to said central processor for storing a number of different process codes, a process scheduling mechanism comprising:

input means coupled to said at least one central processor to receive a command indicating that the process currently being executed by said at least one central processor is in a wait state;

process table means for storing process designations of processes scheduled to be executed by said system including processes which are available to be executed and arranged according to assigned priorities;

output means coupled to said at least one central processor to transmit a move command to move execution to a different process in said at least one memory module; and

control means coupled to said input means, process table means, and output means to cause said output means to transmit said move command along with the process designation of the next highest priority available process;

said process table means having said stored process designations arranged as a linked list of said designations according to said assigned priorities.

8. A process scheduling mechanism according to claim 7 further including:

priority computation means coupled to said process table means and to said control means to change the priority of various process designations in said process table means when new processes are scheduled for execution by said processing system.

9. A process scheduling mechanism according to claim 8 further including logic unit means coupled to said control means and said process table means to change process state designations in said process table means as the various processes change from the being executed state to a wait state or a ready state.

10. A process scheduling mechanism according to claim 9 wherein:

said control means includes a control store to store sets of control signals for transmission to said input means, process table means and output means to effect their operation.

11. A process scheduling mechanism according to claim 10 wherein:

said output means includes a local memory as an output buffer which memory stores said move to new process command.

12. A process scheduling mechanism according to claim 11 wherein said input means further includes:

message buffer means to store a plurality of commands from said at least one of said central processors.

13. In a processing system having at least one central processor and at least one memory module for storing a plurality of processes to be executed by said at least one processor, which processes require different events to occur before their execution can be completed, an event allocation mechanism comprising:

input means coupled to said at least one central processor to receive a command to allocate an event token to a process currently being executed;

event table means coupled to said input means to store status information about various event designations tokens including whether that event token has been allocated and whether that event has occurred;

output means coupled to said event table means and to said at least one central processor for transmission of a requested event token received from said event table means; and

control means coupled to said input means, output means and event table means to decode said command and to maintain the status of the various event tokens including whether they are available.

14. An event allocation mechanism according to claim 13 wherein:

said control means includes a control store for transmission of sets of control signals to effect the operation of the event table means and output means in response to the decoding of a command from said at least one central processor.

15. An event allocation mechanism according to claim 14 wherein:

said input means includes a message buffer coupled to said control store to store additional processor commands for decoding by said control store.

16. An event allocation mechanism according to claim 13 wherein:

said event table means contains a linked list of all event tokens that have been allocated to a particular process.

17. An event allocation means according to claim 13 wherein:

said event table means includes a link table means for storing a list of all processes having requested procurement of each of the particular event tokens.

18. An event allocation mechanism according to claim 13 wherein:

said event table means includes a link table means for storing a list of all processes waiting on each of the allocated event tokens to have occurred.

19. In a processing system having at least one central processor and at least one memory module for storing a plurality of processes to be executed by said at least one processor, which processes require different events to occur before their execution can be completed, an event allocation mechanism comprising:

input means coupled to said at least one central processor to receive a command to allocate an event token to a process currently being executed;

event table means coupled to said input means to store status information about various event tokens including whether that event token has been allocated and whether the event has occurred;

output means coupled to said event table means and to said at least one central processor transmission of a requested event token received from said event table means; and

control means coupled to said input means, output means and event table means to decode said command to maintain the status of the various event tokens including whether they are available;

said control means changing the event status in said event table to has occurred in response to a command received by said input means.

20. An event allocation mechanism according to claim 19 wherein:

said control means include a control store for transmission of sets of control signals to effect the operation of the event table means and output means in response to the decoding of a command from said at least one central processor.

21. An event allocation mechanism according to claim 20 wherein:

said input means includes a message buffer coupled to said control store to store additional processor commands for decoding by said control store.

22. An event allocation mechanism according to claim 21 wherein:

said event table means contains a linked list of all event tokens that have been allocated to a particular process.

23. An event allocation means according to claim 22 wherein:

said event table means includes a link table means for storing a list of all processes having requested procurement of each of the particular event tokens.

24. An event allocation mechanism according to claim 23 wherein:

said event table means includes a link table means for storing a list of all processes waiting on each of the allocated event tokens to have occurred.

25. In a processing system having at least one central processor and at least one memory module for storing a plurality of processes to be executed by said at least one central processor, which processes require different events to occur before their execution can be completed, a task control mechanism comprising:

input means coupled to said at least one central processor to receive a series of commands to allocate an event token to a currently executing process and then to indicate that said currently executing process is in a wait state;

process table means for storing process designations of processes scheduled to be executed by said system including processes which are available to be executed and arranged according to assigned priorities;

event table means coupled to said input means to store status information about various event tokens including whether that token has been allocated and whether that event has occurred;

output means coupled to said process table means, event table means and said at least one central processor to transmit to said at least one central processor an available event token and then a move command to move execution to a different process; and

control means coupled to said input means, output means, process table means and event table means to cause said output means to transmit said requested commands.

26. A task control mechanism according to claim 25 wherein:

said control means includes a control store to store sets of control signals for transmission to said input means, process table means and output means to effect their operation.

27. A task control mechanism according to claim 25 wherein:

said output means includes a local memory as an output buffer which memory stores said move to new process command.

28. A task control mechanism according to claim 25 wherein said input means further includes:

message buffer means to store a plurality of commands from said at least one central processor.

29. A task control mechanism according to claim 25 further including:

logic unit means coupled to said control means and said process table means to change process state designations in said process table means as the various processes change from the being executed state to a wait state or a ready state.

30. A task control mechanism according to claim 25 further including:

priority computation means coupled to said process table means and to said control means to change the priority of various process designations in said process table means when new processes are scheduled for execution by said processing system.

31. A task control mechanism according to claim 25 wherein:

said event table means contains a linked list of all event tokens that have been allocated to a particular process.

32. A task control means according to claim 25 wherein:

said event table means includes a link table means for storing a list of all processes having requested procurement of each of the particular event tokens.

33. A task control mechanism according to claim 25 wherein:

said event table means includes a link table means for storing a list of all processes waiting on each of the allocated event tokens to have occurred.

34. In a processing system having at least one central processor and at least one memory module for storing a plurality of processes to be executed by said at least one central processor, which processes require different events to occur before their execution can be completed, a task control mechanism comprising:

input means coupled to said at least one central processor to receive a series of commands to allocate an event token to a currently executing process and then to indicate that said currently executing process is in a wait state;

process table means for storing process designations of processes scheduled to be executed by said system including processes which are available to be executed and arranged according to assigned priorities;

event table means coupled to said input means to store status information about various event tokens including whether that token has been allocated and whether that event has occurred;

output means coupled to said process table means, event table means and said at least one central processor to transmit to said at least one central processor an available event token and then a move command to move execution to a different process; and

control means coupled to said input means, output means, process table means and event table means to cause said output means to transmit said requested commands;

said process table means having said stored process designations arranged as a linked list of said designations according to said assigned priorities.

35. A task control mechanism according to claim 34 further including:

priority computation means coupled to said process table means and to said control means to change the priority of various process designations in said process table means when new processes are scheduled for execution by said processing system.

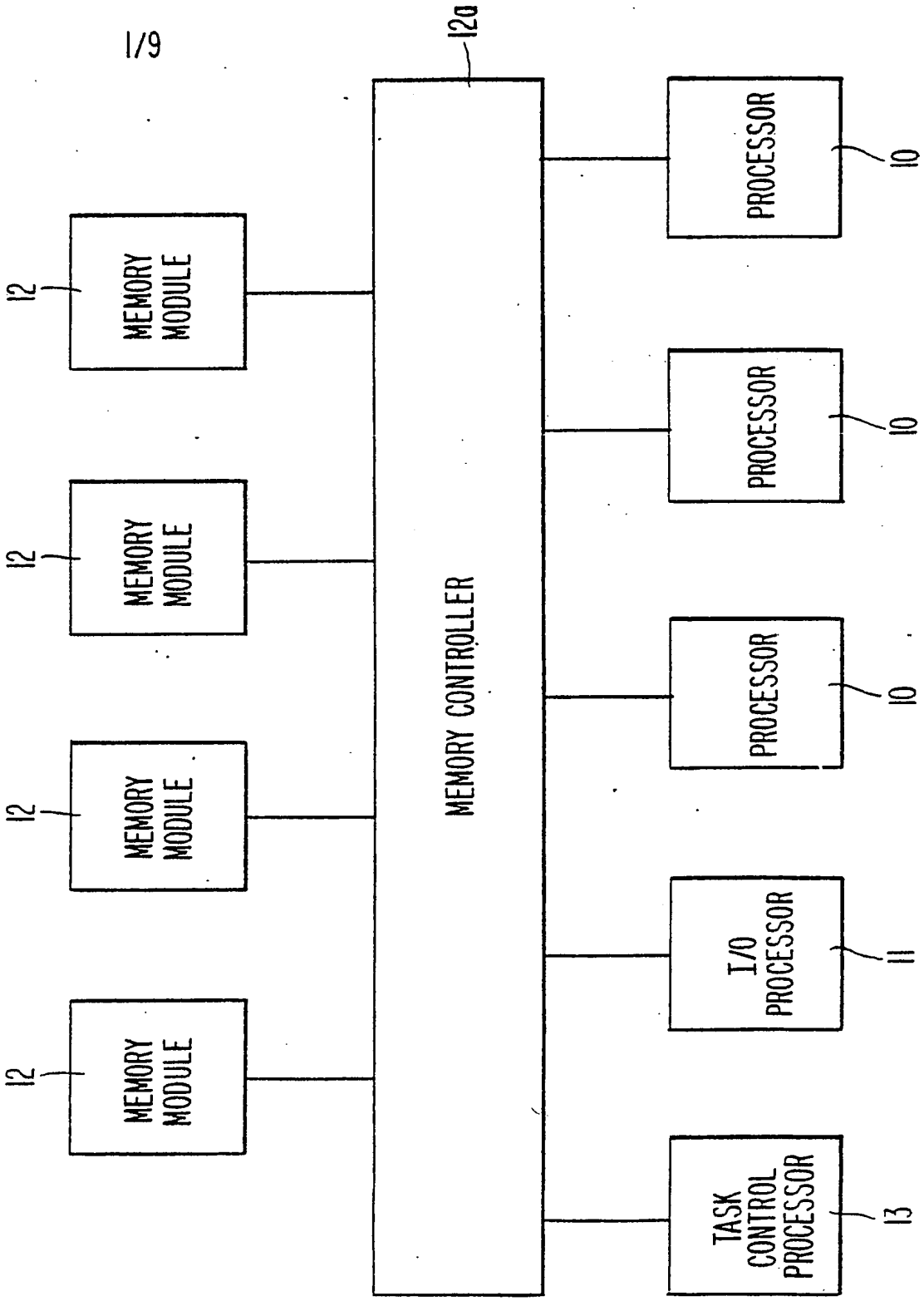

120
MEMORY CONTROLLER
12
MEMORY MODULE
12
MEMORY MODULE
12
MEMORY MODULE
12
MEMORY MODULE
PROCESSOR
10
PROCESSOR
10
PROCESSOR
10
I/O PROCESSOR
11
TASK CONTROL PROCESSOR
13

Fig.1

Fig. 2A

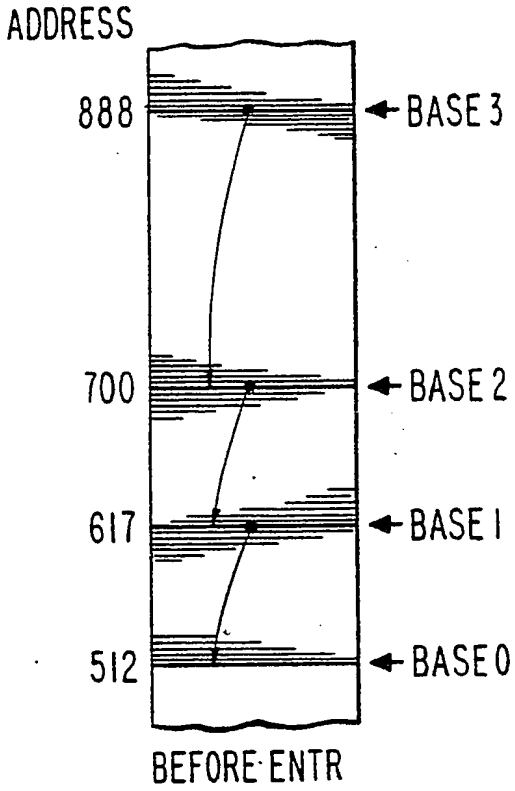


Fig. 2B

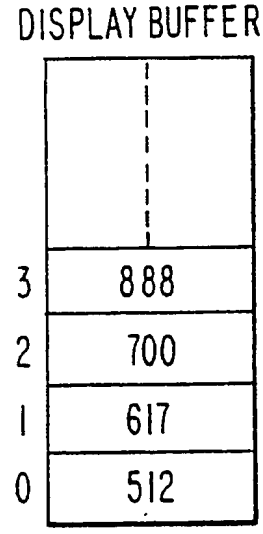


Fig. 2C

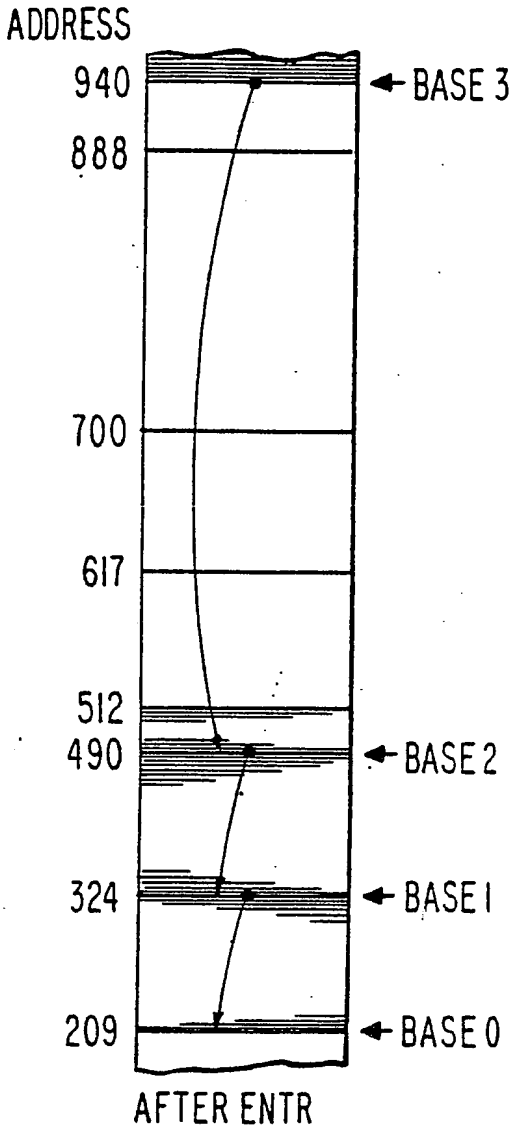
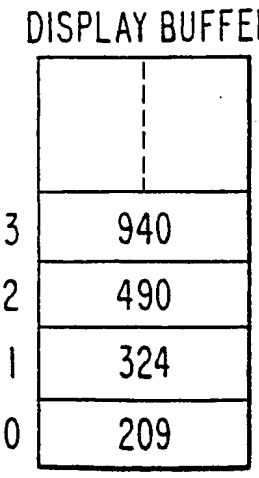


Fig. 2D



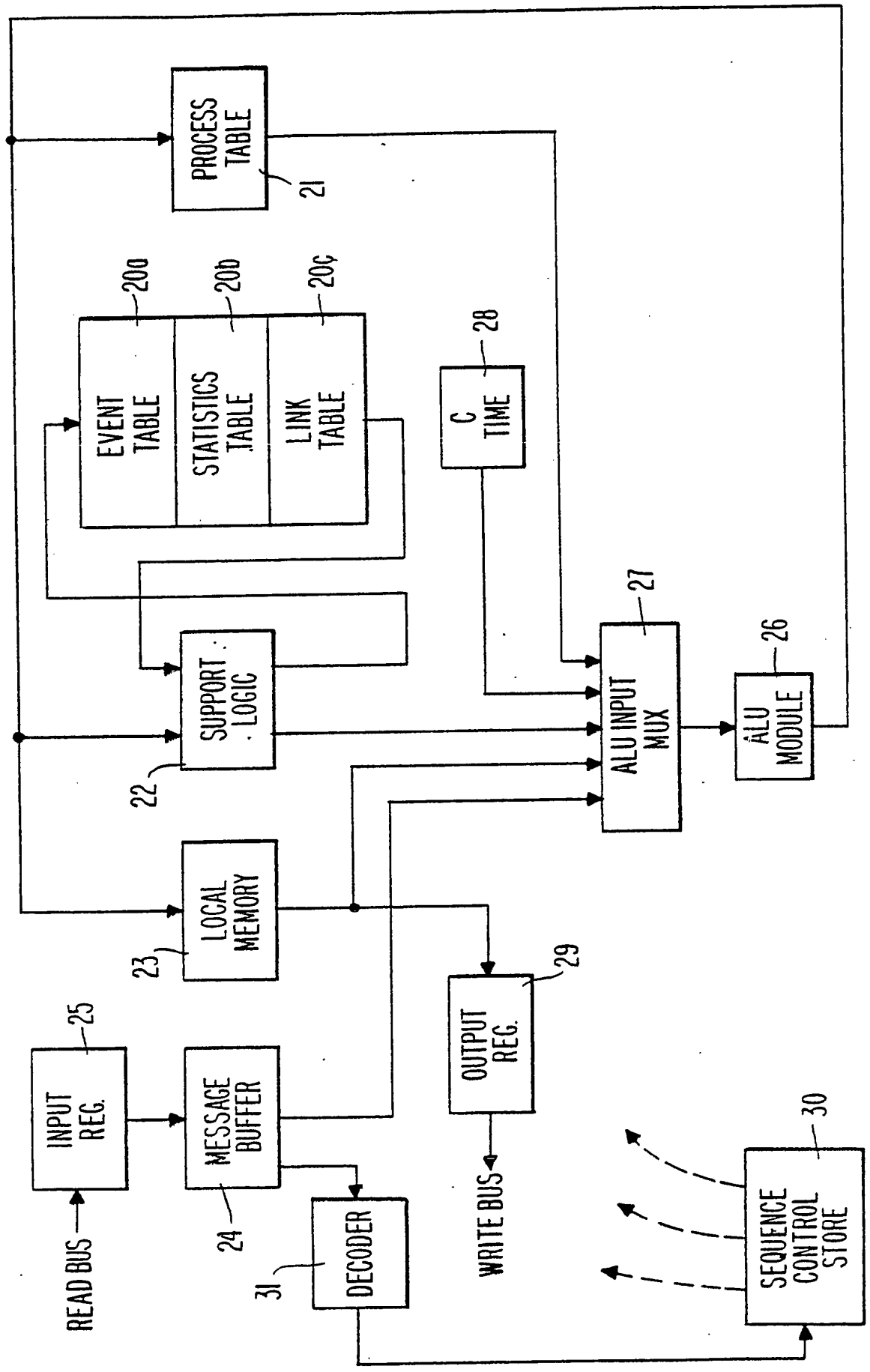

PROCESS TABLE
21
20a
20b
20c
EVENT TABLE
STATISTICS TABLE
LINK TABLE
28
C TIME
SUPPORT LOGIC
22
27
ALU INPUT MUX
26
ALU MODULE
LOCAL MEMORY
23
29
OUTPUT REG.
25
INPUT REG.
MESSAGE BUFFER
24
DECODER
31
READ BUS
WRITE BUS
30
SEQUENCE CONTROL STORE

Fig.3

Fig.4A

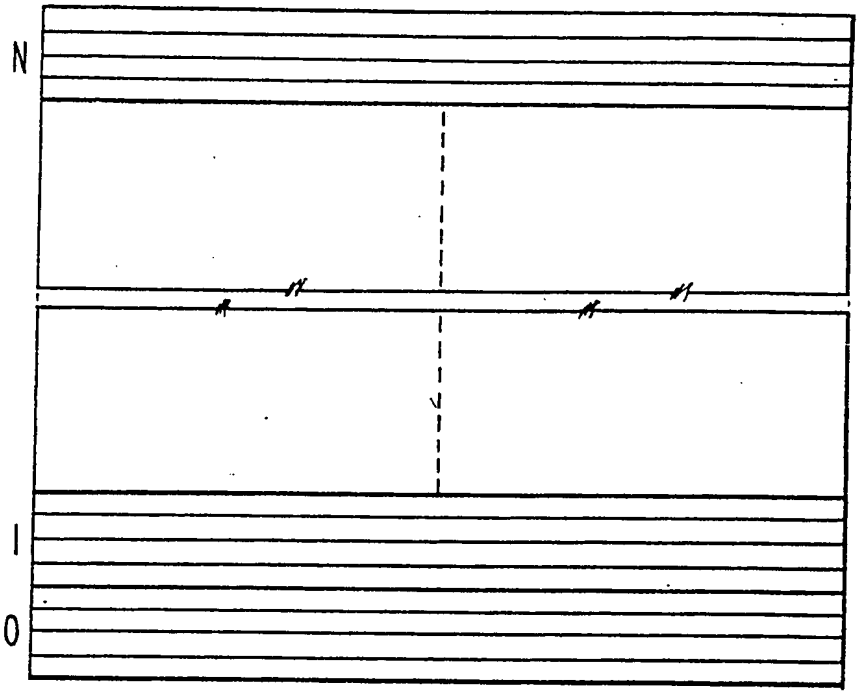
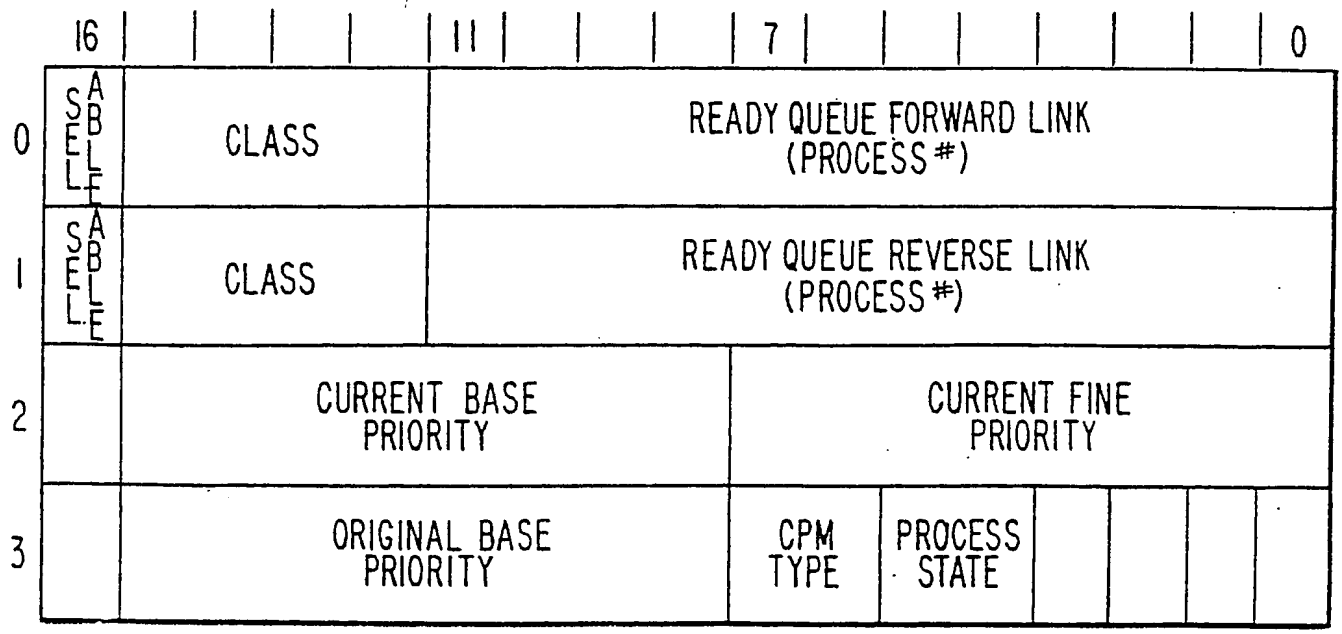

N
1
0

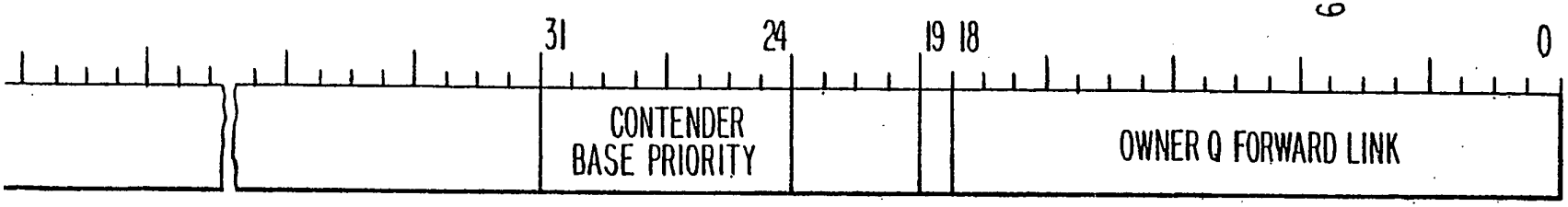
Fig.4B


16
11
7
0
0
SABLE
CLASS
READY QUEUE FORWARD LINK (PROCESS #)
1
SABLE
CLASS
READY QUEUE REVERSE LINK (PROCESS #)
2
CURRENT BASE PRIORITY
CURRENT FINE PRIORITY
3
ORIGINAL BASE PRIORITY
CPM TYPE
PROCESS STATE

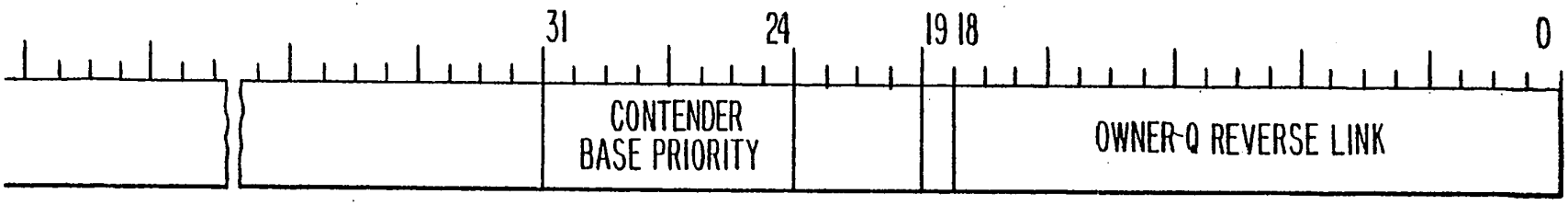
WO 87/02486

PCT/US86/02018

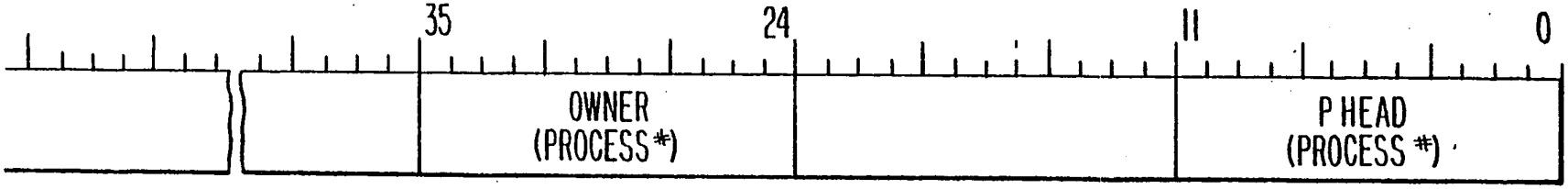
.5A



.5B



.5C



.5D

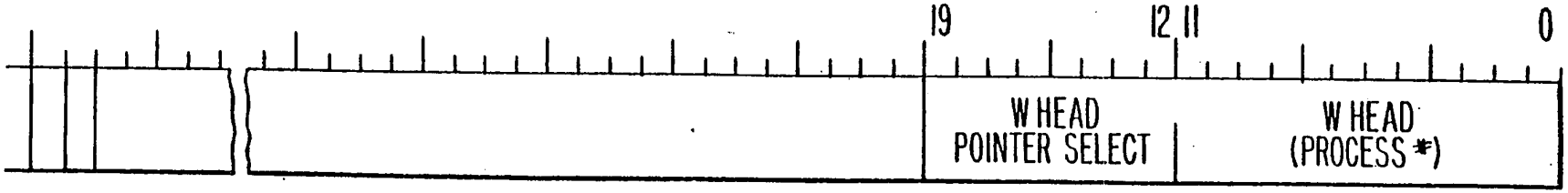
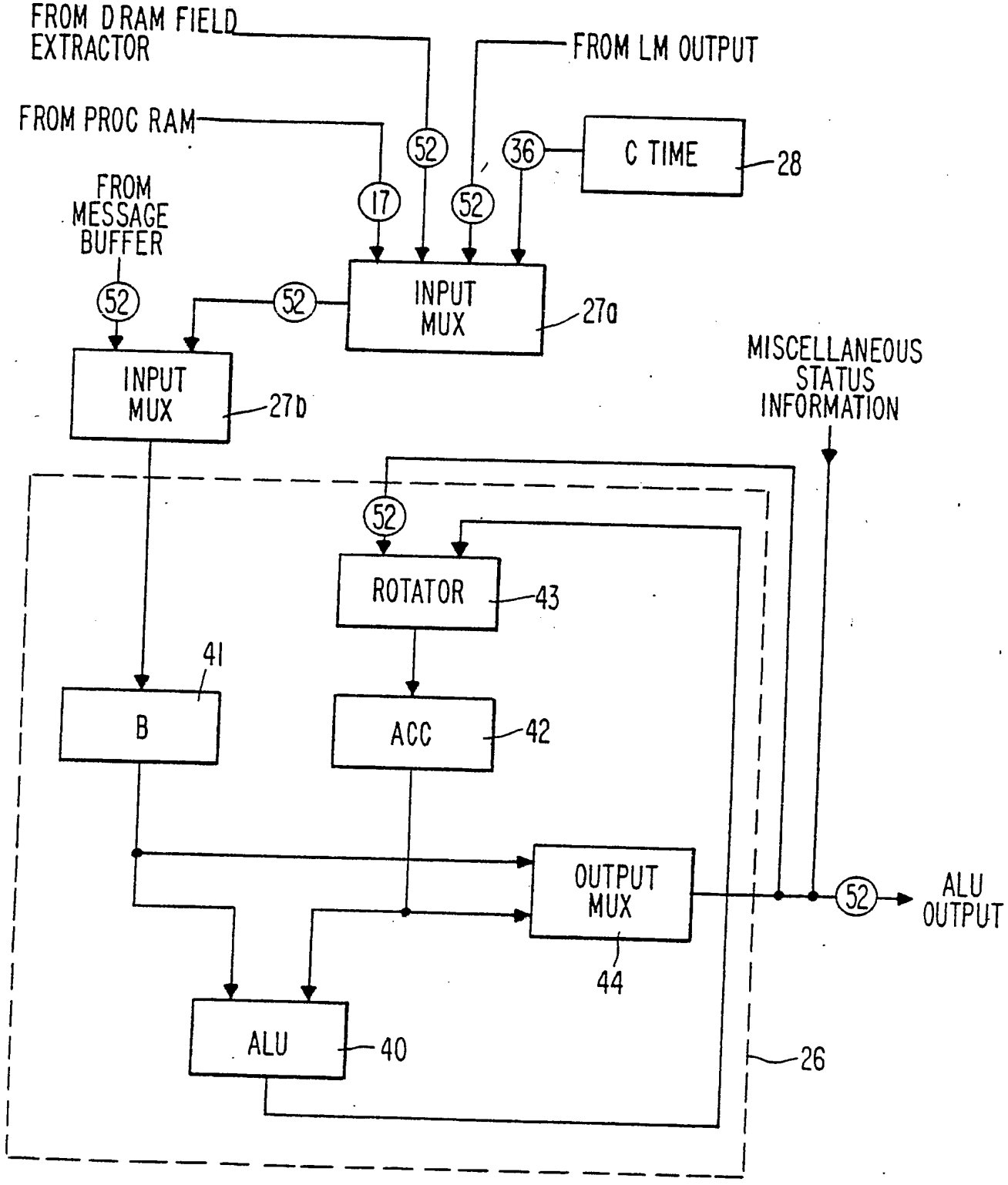


Fig.6


FROM DRAM FIELD EXTRACTOR
FROM LM OUTPUT
FROM PROC RAM
52
36
C TIME
28
FROM MESSAGE BUFFER
17
52
52
INPUT MUX
27a
52
INPUT MUX
27b
MISCELLANEOUS STATUS INFORMATION
52
ROTATOR
43
41
B
ACC
42
OUTPUT MUX
44
52
ALU OUTPUT
ALU
40
26

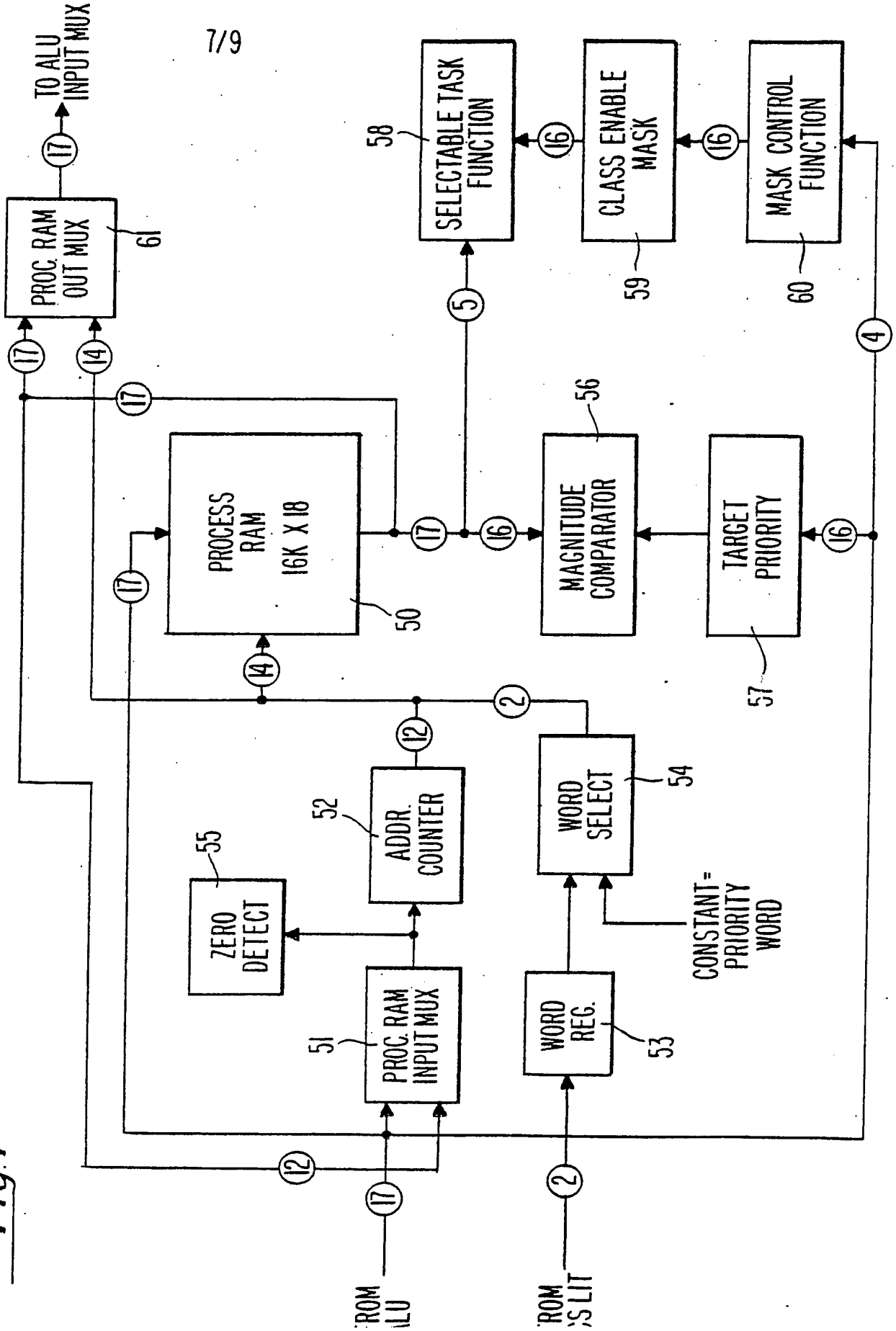

TO ALU INPUT MUX
PROC. RAM OUT MUX
61
58
SELECTABLE TASK FUNCTION
CLASS ENABLE MASK
59
MASK CONTROL FUNCTION
60
PROCESS RAM 16K X 18
50
MAGNITUDE COMPARATOR
56
TARGET PRIORITY
57
ADDR. COUNTER
52
ZERO DETECT
55
PROC. RAM INPUT MUX
51
WORD SELECT
54
WORD REG.
53
CONSTANT= PRIORITY WORD
ROM ALU
ROM S LIT

Fig.7

Fig.8

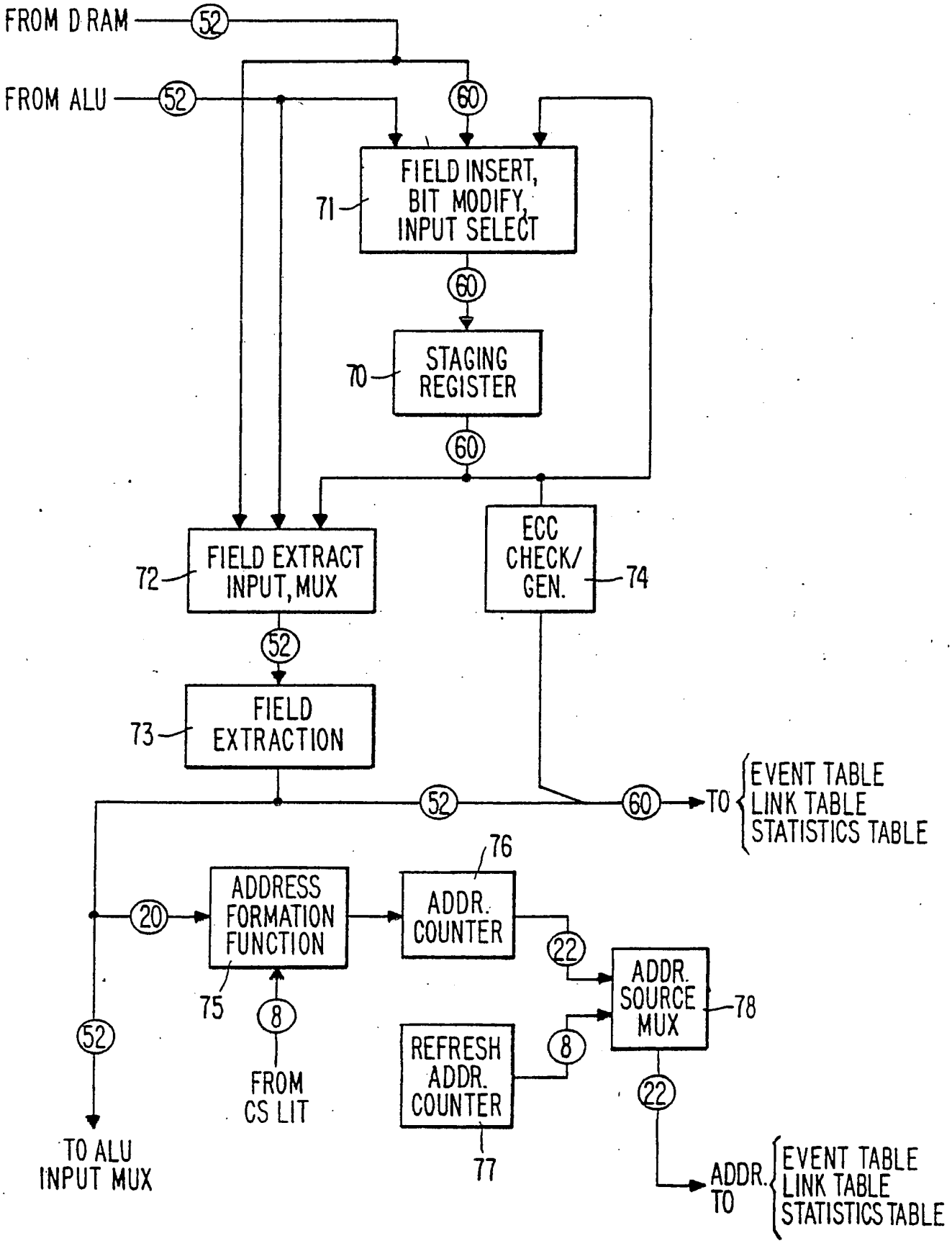

FROM D RAM
52
FROM ALU
52
60
FIELD INSERT, BIT MODIFY, INPUT SELECT
71
60
STAGING REGISTER
70
60
FIELD EXTRACT INPUT, MUX
72
ECC CHECK/ GEN.
74
52
FIELD EXTRACTION
73
52
60
TO
EVENT TABLE
LINK TABLE
STATISTICS TABLE
76
ADDRESS FORMATION FUNCTION
ADDR. COUNTER
20
22
ADDR. SOURCE MUX
78
75
8
52
FROM CS LIT
REFRESH ADDR. COUNTER
8
22
TO ALU INPUT MUX
77
ADDR. TO
EVENT TABLE
LINK TABLE
STATISTICS TABLE

Fig.9A

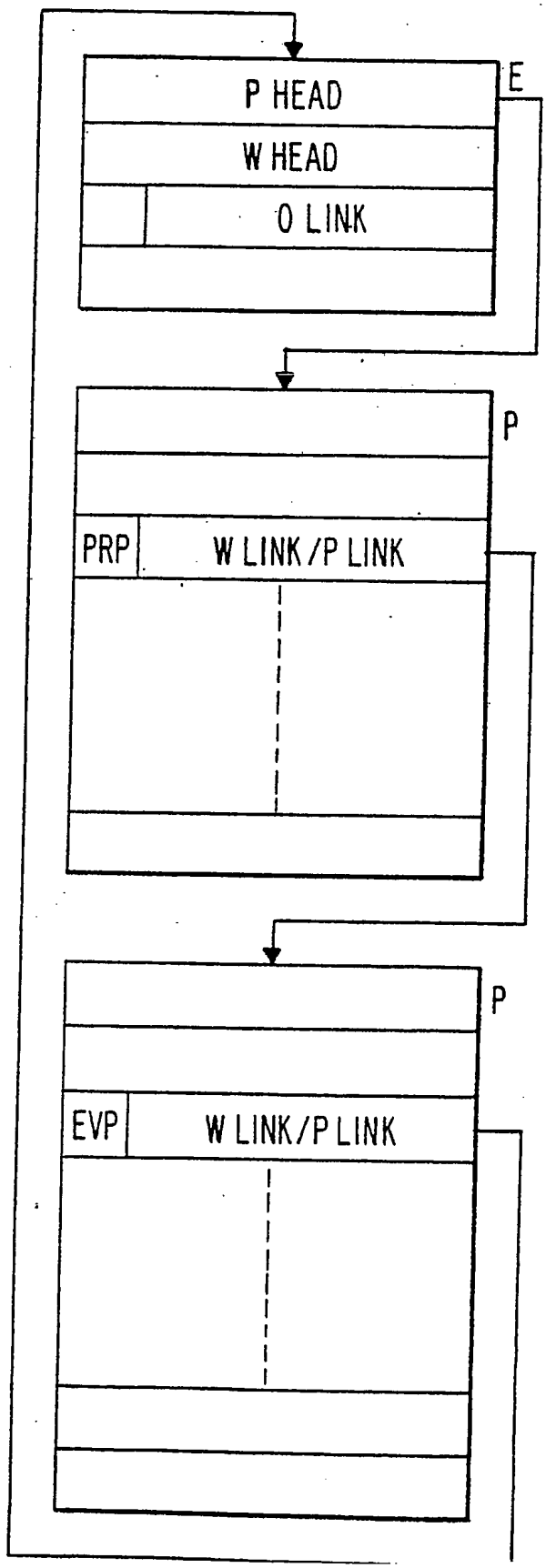
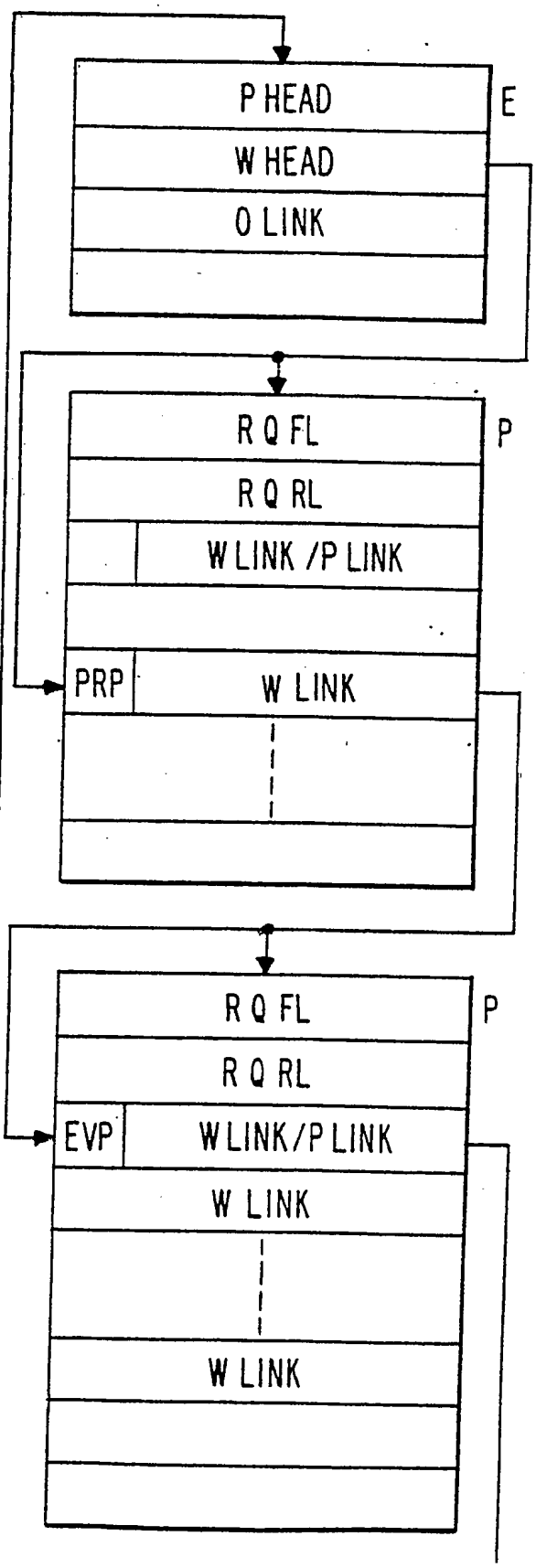

P HEAD
E
W HEAD
O LINK
P
PRP
W LINK/P LINK
P
EVP
W LINK/P LINK

Fig.9B


P HEAD
E
W HEAD
O LINK
R Q FL
P
R Q RL
W LINK /P LINK
PRP
W LINK
R Q FL
P
R Q RL
EVP
W LINK/P LINK
W LINK
W LINK

INTERNATIONAL SEARCH REPORT

International Application No PCT/US 86/02018

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [5]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC[4]: G 06 F 9/46

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
IPC[4]	G 06 F 9/46

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [9]

Category *	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No. [13]
A	Communications of the A.C.M., volume 24, no. 10, October 1981, (New York, US), Denning et al.: "Low contention semaphores and ready lists", pages 687-699 see the whole article	1,2,5,7, 13,14,16-19,22-25, 29,31-34
A	IEEE Transations on Computers, volume C-33, no. 7, July 1984, (New York, US), Männer: "Hardware task/processor scheduling in a polyprocessor environment", pages 626-636 see page 627, left-hand column, lines 3-6; page 628, right-hand column, line 9 - page 630, left-hand column, line 24	1,2,7,13, 19,25,34
A	EP, A2, 0064142 (HOFFMAN, IBM) 10 November 1982 see page 1, line 9 - page 2, line 11	1,7,13,19, 25,34

* Special categories of cited documents: [10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
12th January 1987	12 FEV. 1987
International Searching Authority	**Signature of Authorized Officer**
EUROPEAN PATENT OFFICE	M. VAN MOL

ANNEX TO TH NTERNATIONAL SEARCH REPO[redacted]ON

INTERNATIONAL APPLICATION NO. PCT/US 86/02018 (SA 14799)

This Annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report. The members are as contained in the European Patent Office EDP file on 28/01/87

The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A- 0064142	10/11/82	JP-A-	57187759	18/11/82
		US-A-	4394727	19/07/83
		CA-A-	1173971	04/09/84

For more details about this annex :
see Official Journal of the European Patent Office, No. 12/82